Some Disturbing Facts about Depth First Left Most Variable Ordering Heuristics for Binary Decision Diagrams

Antoine Rauzy
IML/CNRS
163, Avenue de Luminy
13288 Marseille Cedex 9
FRANCE
arauzy@iml.univ-mrs.fr

Abstract: Binary Decision Diagrams have proven to be a very efficient tool to assess Fault Trees. However, the size of the Binary Decision Diagram, and therefore the efficiency of the whole methodology, depends dramatically on the choice of variable ordering. The determination of the best variable ordering is intractable. Therefore, heuristics have been designed to select reasonably good variable orderings. The most popular of these heuristics consists in numbering variables by means of a depth first left most traversal of the formula, after possibly some rearrangements of inputs of gates. In this article, this heuristic is studied in a systematic way. It is shown to be very sensitive to the way the formula is written. A series of experiments shows also that the notion of locality of variables, which was believed to be a key issue in the determination of good orderings, must be handled with care. These facts are quite disturbing and raise questions about the design of good variable ordering heuristics.

Keywords: Fault Trees, Binary Decision Diagrams, Variable Ordering Heuristics

1. Introduction

Since their introduction in the Reliability Engineering field, Bryant's Binary Decision Diagrams [**1**, **2**] have proved to be a very efficient tool to assess Fault Trees [**3**, **4**]. It is well known however that the size of the Binary Decision Diagram, and therefore the efficiency of the whole methodology, depends dramatically on the choice of variable ordering [**1**, **5**]. The determination of the best variable ordering is intractable [**6**, **7**]. Therefore, heuristics have been designed to select reasonably good ones. There are two kinds of heuristics: static heuristics, that number the variables prior to the BDD computation (see e.g. [**8**, **9**]), and dynamic heuristics that reorder the BDD during the computation (see e.g. [**10**, **11**]). Dynamic heuristics are very time consuming, especially when the number of variables is large, which is the case for real world Fault Tree/Event Tree models. So, even if such a heuristic is used, it is best to start with an acceptable order given by some static heuristic. Static heuristics are based on topological considerations (sizes of the sub-formulae, depths of the variables, weights…). They can be in turn separated into two categories [**12**, **13**]: those that number variables according to a depth first left most (DFLM) traversal of the formula, after possibly some rearrangements of inputs of gates, and those that build interleaved orders. In this article, we study in a systematic way heuristics of the first category, so-called DFLM heuristics in the sequel. In practice, these heuristics give rather good results. They have never been outperformed by any other method proposed in the literature. These good performances stand partly in the fact that they tend to preserve the locality of variables: variables that are close in the formula tend to be not too far apart in the ordering. However, this is not completely true and that the notion of locality itself is questionable.

For the purpose of this study, a benchmark of twenty real life large fault trees has been selected. These fault trees come from different sources (nuclear industry, avionic systems…) and have different characteristics (number of variables, number of gates, shape of the formula…). In

order to make experiments both of interest and tractable, only trees for which the BDD, computed with a pure DFLM heuristic, is not too small and not too large (typically few hundred thousands nodes) have been included in the benchmark . For each of these trees, one hundred different formulae by shuffling at random inputs of gates have been derived. These formulae are thus strictly the equivalent to the original tree up to an isomorphism. The idea is to evaluate the sensitivity of the different heuristics to the way the formula is (re)written. It is worthwhile to notice that, due to the use of graphical user interfaces and/or automatic generation of formulae, the user of a fault tree assessment tool has in general no control on the order of inputs of a gate (and in any case, no idea of what a good order would be).
The contribution of this article is the following: a first series of experiments shows that the pure DFLM ordering is subject to incredibly high variations of in terms of sizes of BDD and total numbers of BDD nodes involved during the computation. This problem is almost completely ignored in the literature (except a previous article by the author [**14**]). A second series of experiments is devoted to heuristics that sort inputs of gates according to some weighting algorithm (e.g. [**9**, **15**]). These experiments show that weighting heuristics may lead to far from optimal results, although they are able to reduce the instability. Finally, a third series of experiments shows the lack of correlation between the BDD sizes and the linear arrangement of gates and basic events. The idea to optimize linear arrangement to improve variable ordering is rather natural (see e.g. [**16**, **8**, **17**]). Unfortunately, it is quite deceptive.
The remainder of this article is organized as follows. Section 2 recalls basics about assessment of fault trees by means of binary decision diagrams. It presents also our benchmark. Section 3 reports results of the pure DFLM heuristic. Section 4 discusses weighting heuristics. Section 5 studies the correlation between the size of the BDD and the linear arrangement of gates and basic events.

2. Fault Trees Assessment by means of Binary Decision Diagrams

1.1. Fault Trees and Depth First Left Most traversals

In this article, (coherent) fault trees built over basic and intermediate events and 'and', 'or' and 'k-out-of-n' gates are considered. It is assumed as usual that the fault tree is rooted by a top event. Moreover, it is assumed, without a loss of generality, that each gate is named with an intermediate event. Therefore, the fault tree structure can be seen (disregarding the type of the gates) as a directed acyclic graph, as exemplified Figure 1.
A depth first left most traversal of the fault tree (or equivalently of the underlying graph) produces an order on the basic events. For instance, in the tree pictured Figure 1, basic events and gates are visited in the following order: top, g1, e1, g3, e2, e3, g2, g3 (bis), g4, e2 (again) and e4. Therefore, the basic event ordering is e1<e2<e3<e4. By rearranging inputs of gates, a different ordering can be produced. E.g. if e1 and g3 are switched under g1, the order is e2<e3<e1<e4. Not all of the (n! for n basic events) orderings can be produced. For instance, the ordering e1<e4<e2<e3 cannot, whichever rearrangement of inputs of gates is considered. Moreover, two different writings may produce the same ordering. For instance, switching g3 and g4 under g2 in the original formula has no effect on the ordering of basic events.

1.2. The Benchmark

For the purpose of this study, twenty real life large fault trees as a benchmark have been selected. These fault trees come from different sources (nuclear industry, avionic systems…). Their number of variables and number of gates are given Table 1.
To rearrange at random inputs of gates, the method is the following. First, a unique index is associated at random with each gate and basic event. Then, inputs of gates are sorted according to these indices. This procedure ensures that if two variables (gates or basic events) appear

under two different gates, then they appear in the same order. This corresponds certainly to what would do a human or a program. This procedure is also faster than to shuffle inputs of gates independently.

1.3. Binary Decision Diagrams

Binary Decision Diagrams are a compact encoding of the truth tables of Boolean formulae [**1**, **2**]. The BDD representation is based on the Shannon decomposition: Let F be a Boolean function that depends on the variable *v*, then the following equality holds.

$$F = v.F[v \leftarrow 1] + \bar{v}.F[v \leftarrow 0]$$

By choosing a total order over the variables and applying recursively the Shannon decomposition, the truth table of any formula can be graphically represented as a binary tree. The nodes are labeled with variables and have two out edges (a *then*-out edge, pointing to the node that encodes F[v←1], and an *else*-out edge, pointing to the node that encodes F[v←0]). The leaves are labeled with either 0 or 1. The value of the formula for a given variable assignment is obtained by descending along the corresponding branch of the tree. The Shannon tree for the formula $F = ab + \bar{a}c$ and the lexicographic order is pictured Figure 2 (dashed lines represent *else*-out edges).

Indeed such a representation is very space consuming. It is however possible to shrink it by means of the following two reduction rules.

- Isomorphic sub-trees merging. Since two isomorphic sub-trees encode the same formula, at least one is useless.
- Useless nodes deletion. A node with two equal sons is useless since it is equivalent to its son ($F = v.F + \bar{v}.F$).

By applying these two rules as far as possible, one get the BDD associated with the formula. A BDD is therefore a directed acyclic graph. It is unique, up to an isomorphism. This process is illustrated Figure 2. The reader is invited to refer to the wide literature of BDD for a more detailed presentation of this technique.

1.4. Variable Ordering Heuristics and Pre-processing of the Formulae

As said in the introduction, static heuristics rely in general on topological considerations and depend on the way formulae are written. Rewriting procedures can thus be used both to simplify the trees and to get better variable orderings. The two issues are strongly related. Any complete fault tree analysis BDD strategy should involve such rewritings, as illustrated in references [**18**, **19**]. Nevertheless, this article focuses on only one type of rewriting: the rearrangement of inputs of gates, which deserves itself a systematic study.

As an illustration, consider again the fault tree pictured Figure 1. Since there are 4 basic events, there are 4! = 24 possible variable orderings. However, only 8 are reachable through DFLM heuristics. Figure 3 shows the corresponding rewritings and gives the sizes to the BDD of the top event as well as the number of nodes involved during the computation. This example illustrates these two quantities are not necessarily correlated. Both are of interest: the smaller the BDD, the easier the subsequent computations (probabilities, importance factors, minimal cut sets…). On the other hand, to be feasible, the computation of the BDD should not involve too many intermediate nodes. The best order is obtained by putting the variable e2 in front. This variable is actually the one which most influences the formula. Heuristics aim to put the most influential variables in front, while keeping related variables together (here e2 and e3).

Another example, first proposed by Bryant himself [**5**], illustrates the importance of variable ordering. It helps also to understand why the DFLM heuristics are not too bad in general. Let $F_n = x_1.y_1 + x_2.y_2 + \ldots + x_n.y_n$. There are two natural orderings for this formula: the lexicographic order $x_1<x2<\ldots<x_n<y_1<y_2<\ldots<y_n$ and the ordering obtained by means of a DFLM traversal of the formula, i.e. $x_1<y_1<x_2<y_2<\ldots<x_n<y_n$. The BDD for these two orderings

are pictured Figure 4 for the case n=3. The DFLM ordering gives a linear size BDD, while the lexicographic ordering gives an exponential size BDD! In the first case, the BDD is in O(n) because once the value of $x_1.y_1$ is determined, either F_n is satisfied or it is reduced to $x_2.y_2 + \ldots + x_n.y_n$, which is isomorphic to F_{n-1}. The DFLM heuristic preserves the locality of variables. The lexicographic ordering gives an $O(2^n)$ BDD because before deciding the value of the formula, the value of all of the x_i's must be considered. Therefore the top part of the BDD is a complete binary tree, as shown Figure 4. It does not preserve locality.

3. Sensitivity to Ordering of Inputs of Gates of the DFLM heuristic

Reference [**14**] pointed out the sensibility of the pure DFLM heuristic to rearrangements of inputs of gates. The models for this study were mostly coming from French nuclear PSA. In order to confirm these preliminary results with our broader benchmark, the pure DFLM heuristic has been applied on each of the 100 formulae derived from each tree. The minimum, maximum and mean BDD sizes and numbers of nodes involved during the computation are given Table 2. For the sake of the clarity, maximum and mean values are written as a multiple of the minimum value (over the 100 random rewritings).

This experiment shows that the pure DFLM heuristic is dramatically sensitive to the way the formula is written: the factor between the minimum and the maximum (respectively mean) BDD size goes up to 128 (respectively 17). The factor between the minimum and maximum (respectively mean) numbers of nodes involved during the computation goes up to 81 (respectively 10). This makes the comparison of this heuristic (applied to the original formula) with any other heuristics very questionable: a simple switch of some inputs of some gates may change completely the conclusion of the comparison. Many articles, however, (including, the author must admit, his own) take this heuristic as a comparison basis. This experiment is a strong argument to definitively reject this approach.

4. Weighting Heuristics

Many heuristics proposed in the literature can be reinterpreted as three steps procedures: first, a weight is associated with each variable (gates and basic events); second, inputs of gates are sorted according to this weight (in increasing or decreasing order); third the pure DFLM heuristic is applied to the resulting formula. For instance, the heuristic proposed in references [**9**, **20**] consists in defining the weight of a variable as the number of occurrences of this variable and then in sorting inputs of gates in decreasing order of their weight. The heuristic proposed in reference [**15**] consists in defining the weight of variables as follows.

– The weight of a basic event is 1.0.
– The weight of a gate equals the sum of the weights of its inputs.

Then, inputs of gates are sorted in increasing order of their weights.

For the purpose of this study, many different weighting heuristics have been tried. Only two of them are discussed here, for they are representative.

The first one, so-called DDS (for Decreasing Downward Sums) consists in defining the weight of the top event as 1.0 and then the weight of any other variable as the sum of the weights of its outputs. Inputs of gates are sorted in decreasing order of their weights. The heuristic DDS generalizes the underlying idea of the heuristic proposed in reference [**9**]: it looks for most important variables.

The second one, so-called DUS (for Decreasing Upward Sums) consists in defining the weights of variables in the same way the heuristic proposed in reference [**15**] does, but in sorting inputs of gates in decreasing order of their weights. In our experience, the former gives much better results than the latter.

Results for these two heuristics are given respectively Table 3 and Table 4. The ratio between the observed value and the minimum value obtained with the pure DFLM heuristic are

indicated (this makes comparisons easier than with raw numbers). For instance, the value 3.1 in "abs-001" row and mean |BDD| column of Table 3 means that the mean BDD size observed for the tree "abs-001" with heuristic DDS is 3.1 times bigger than the minimum BDD size observed with the pure DFLM heuristic.
Compared to the pure DFLM heuristics, DDS reduces a bit the instability. In some cases, it improves slightly minimum, maximum and mean values as well. In some other cases, it degrades minimum and mean values. It is worth to note that DDS does not reorder the top gates, for weights of these gates are all equal to 1. Therefore it has only a relatively small influence on the variable ordering (which explains that it is very comparable to the pure DFLM).
With DUS on the other hand, the instability almost vanish. But for most of the cases, results are rather far from the optimum.
This experiment summarizes what can be expected from weighting heuristics: either they are loose, like DDS, and give results that very similar to those of the pure DFLM heuristic or they are strict and they give results that are not near the optimum. This explains why, despite of the wide efforts that have been done to find the "good" weighting procedure, no article in the literature proposes a weighting heuristic that performs consistently better than the pure DFLM heuristic.

5. Correlation with Linear Arrangement

As pointed out by Bryant's example (see section 1.4), variables that are close in the formula should better be close in the order. This remark has been explored by Panda and Somenzi [**11**] and more recently by Aloul & al who proposed several heuristics [**16**, **8**, **17**]. Intuitively, their idea is to associate an index with each variable of the formula, i.e. with each vertex of the underlying graph, and then to use the network width as a measure of the quality of a rewriting. The graph width is defined as the sum over the edges of the distances between adjacent vertices. For instance, in the graph of Figure 1, the graph width W is as follows. W = d(top,g1) + d(top,g2) + d(g1,e1) + d(g1,g3) + d(g2,e3) + d(g2,g4) + d(g3,e2) + d(g3,e3) + d(g4,e2) + d(g4,e3) where d(a,b) = |index(a)-index(b)|. This idea is seductive. Unfortunately, it raises two main difficulties. The first one stands in the numbering itself: if indices are constrained to be all different, finding the variable ordering that minimize the graph width is a well known NP-complete problem, so called Optimal Linear Arrangement [**21**]. Optimal Linear Arrangement is even considered as a hard problem among the NP-complete problems [**22**]. So we may sail from Charybdis to Scylla. The second difficulty stands in the correlation between the graph widths and the sizes of BDD.
In order to verify whether sizes of BDD and graph widths are actually correlated, graph widths for each of the 100 rewritings of the fault trees of the benchmark have been calculated and compared with the size of the BDD for the DFLM basic event ordering. In some sense, this experiment is dual from the approach proposed by Aloul & al.: rather than to induce a good ordering from a good linear arrangement, it is tried to see whether good (respectively bad) variable orderings correspond to good (respectively bad) linear arrangements.
Here is a technical problem: BDD require ordering basic events while linear arrangement requires ordering basic events and gates. The two following methods to get an ordering of gates from the (DFLM) ordering of basic events have been applied.

- As Soon As Possible (ASAP) ordering: the formula is traversed in a depth-first left-most way and basic events and gates are numbered as soon as possible, i.e. a gate is numbered as soon as all of its inputs are numbered. For instance, for the formula pictured Figure 1, the ASAP ordering is as follows: e1, e2, e3, g3, g1, e4, g4, g2, top.
- Mean of Inputs (MI) ordering: first basic events are ordered with the DFLM heuristic, and then each gate is weighted as the mean value of the weights of its inputs (weights

are not indices, but this doesn't change the definition of graph width). For instance, for the formula pictured Figure 1, the first step gives the following weights to basic events.

w(e1) = 1, w(e2) = 2, w(e3) = 3, w(e4) = 4

The second step gives the following weights to gates:

w(g3) = 2.5, w(g4) = 3, w(g1) = 1.75, w(g2) = 2. 75 and w(top) = 2.25

Figure 5 and Figure 6 plot, for each of the fault tree of the benchmark, the points formed by size of the BDD (x-axis) and the value of the graph width (y-axis). Both quantities are plotted with a linear scale.

No matter what are the absolute values of these quantities, these curves show clearly that no correlation can be expected between the size of the BDD for the DFLM ordering and the value of the graph width (measured with either ASAP or MI weighting methods). In other words, the heuristics proposed by Aloul & al may be good, but not for the reason these authors give (by the way, their heuristics did not give especially good results on the formulae of the benchmark).

6. Conclusion

In this article, DFLM variable ordering heuristics have been studied in a systematic way. An experimental study raised different disturbing issues:

- The pure DFLM heuristic is so sensitive to the way the formula is written that it cannot be taken as a reference, at least if applied only on the original writing of the formula.
- Weighting heuristics are either stable and mediocre, or fail into the same instability as the pure DFLM heuristic.
- The notion of graph width does not seem to be correlated with the size of the BDD.

In a word, this study raises more problems than it brings solutions. However, it paves the way for future researches in the domain.

First, this study changes the perspective on heuristics: rather than seeking for good heuristics by mixing different topological measures, it would be of interest to try to understand why an ordering is good or bad. This raises the question of whether there exist measures (other than graph width) that show some correlations with the size of BDD.

Second, it shows that heuristics should not be considered in isolation, but rather as elements of strategies. A strategy consists in applying one or several heuristics on different rewritings of the formula under study, possibly with evolving time and space limitations (as already suggested in reference [**14**]).

7. References

1 **Bryant, R**. Graph Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

2 **Brace K., Rudell R.,** and **Bryant R**. Efficient Implementation of a BDD Package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 40–45. IEEE 0738, 1990.

3 **Coudert, O.** and **Madre, J.-C.** Fault Tree Analysis: 10^{20} Prime Implicants and Beyond. In *Proceedings of the Annual Reliability and Maintainability Symposium, ARMS'93*, January 1993. Atlanta NC, USA.

4 **Rauzy, A.** New Algorithms for Fault Trees Analysis. *Reliability Engineering & System Safety*, Volume 59, Issue 2, pp 203–211, 1993.

5 **Bryant, R**. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys*, 24:293–318, September 1992.

6 **Friedman, S.J.** and **Supowit, K.J**. Finding the Optimal Variable Ordering for Binary Decision Diagrams. *IEEE Transactions on Computers*, 39(5):710–713, May 1990.

7 **Bollig, B.** and **Wegener, I**. Improving the Variable Ordering of OBDDs is NP-Complete. *IEEE Trans. on Software Engineering*, 45(9):993–1001, September 1996.

8 **Aloul, F.A. Markov, I.L.** and **Sakallah K.A**. FORCE: A Fast and Easy-To-Implement Variable-Ordering Heuristic, *Proceedings of Great Lakes Symposium on VLSI* (GLSVLSI), Washington D.C., pp. 116-119, 2003.

9 **Fujita, M. Fujisawa, H.** and **Kawato, N**. Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams. In *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'88*, pages 2–5, 1988.

10 **Rudell, R.** Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'93*, pages 42–47, November 1993.

11 **Panda, S.** and **Somenzi, F.** Who Are the Variables in Your Neighborhood. In *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'95*, pages 74–77, 1995.

12 **Nikolskaia, M.** and **Rauzy, A.** Fine-tuning of Boolean formulae preprocessing techniques. In *Proceedings of the European Safety and Reliability Association Conference, ESREL'99*, volume 2, pages 1027–1032. A.A. Balkema, 1999. ISBN 90 5809 111 2.

13 **Ibañez Llano, C. Meléndez Asensio, E.** and **Nieto Fuentes, F.** Variable ordering schemes to apply to the binary decision diagram methodology for event tree sequences assessment. In *Proceeding of the ESREL'06 Conference*, C. Guedes Soares and E. Zio ed., Taylor & Francis group, ISBN 0-0415-41620-5. pp 1645-1652. 2006.

14 **Bouissou, M. Bruyère, F.** and **Rauzy, A.** BDD based Fault-Tree Processing: A Comparison of Variable Ordering Heuristics. In C. Guedes Soares, editor, *Proceedings of European Safety and Reliability Association Conference, ESREL'97*, volume 3, pages 2045–2052. Pergamon, 1997. ISBN 0-08-042835-5.

15 **Minato, S. Ishiura, N.** and **Yajima, S.** Shared Binary Decision Diagrams with Attributed Edges for Efficient Boolean Function Manipulation. In L.J.M Claesen, editor, *Proceedings of the 27th ACM/IEEE Design Automation Conference, DAC'90*, pages 52–57, June 1990.

16 **Aloul, F. Markov, I.** and **Sakallah, K.** Improving the Efficiency of Circuit-to-BDD Conversion by Gate and Input Ordering. *International Conference on Computer Design* (ICCD), Freiburg, Germany, pp. 64-69, 2002.

17 **Aloul, F. Markov, I.** and **Sakallah, K.** MINCE: A Static Global Variable-Ordering Heuristic for SAT Search and BDD Manipulation. *Journal of Universal Computer Science* (JUCS), 10(12), pp. 1562-1596, Dec. 2004

18 **Reay, K.A.** and **Andrews, J.D.** A fault tree analysis strategy using binary decision diagrams, R*eliability Engineering and System Safety*. Volume 78, Issue 1, pp 45-56, October 2002.

19 **Epstein, S.** and **Rauzy, A.** Can We Trust PRA ?, *Reliability Engineering and System Safety*, Volume 88, Issue 3, pp 195-205, 2005.

20 **Fujita, M. Fujisawa, H.** and **Matsugana, Y.** Variable Ordering Algorithm for Ordered Binary Decision Diagrams and Their Evalutation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(1):6–12, January 1993.

21 **Garey, M.R.** and **Johnson, D.S.** Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Fransisco, 1979.

22 **Horton, S.B.** *Optimal Linear Arrangement problem: algorithms and approximation*. Thesis, Georgia Institute of Technology, May, 1997.

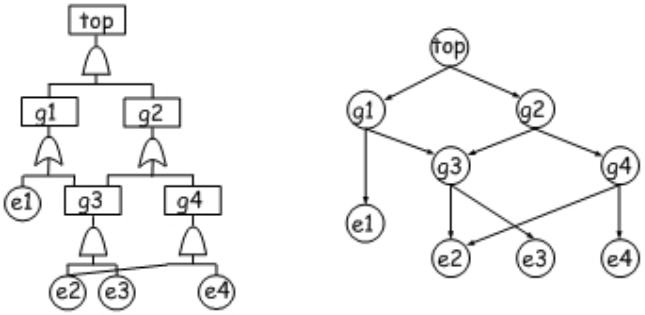

Figure 1. A fault tree and its underlying acyclic graph.

Table 1. Number of basic events and gates of the fault trees of the benchmark

Fault Tree	#BE	#gates
abs-001	763	934
abs-002	177	130
abs-007	310	115
cpf-001	127	81
das-010	122	288
edf-003	362	475
edf-004	323	375
edf-008	278	255
edf-010	311	290
edf-011	283	249
elf-001	145	242
jbd-001	533	315
las-001	424	371
las-003	461	387
las-005	462	385
mhi-301	505	285
mhi-302	890	
mhi-303	614	363
par-001	409	493
par-006	345	475
sof-001	186	1072

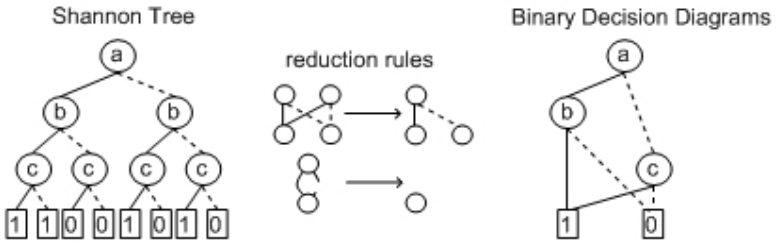

Figure 2. From the Shannon Tree to the BDD

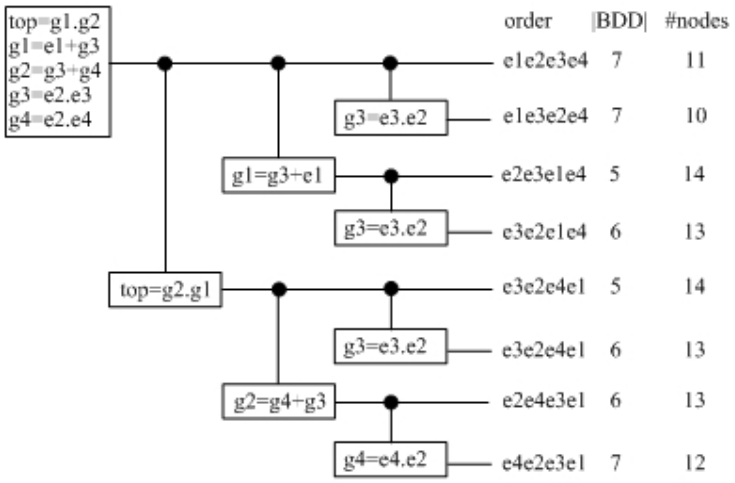

Figure 3. Effects of different rewritings on variable ordering

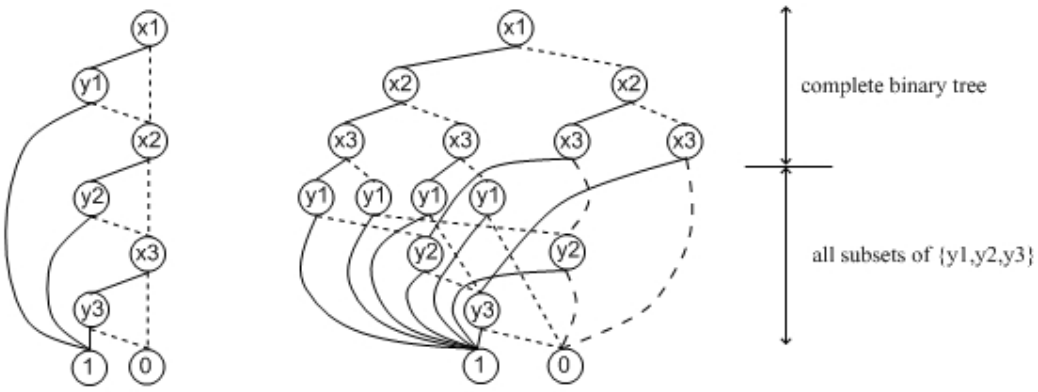

Figure 4. BDD for two different variable orderings

Table 2. Heuristic DFLM: BDD sizes and total numbers of nodes involved in the computation

	|BDD|			#nodes	
model	min	max	mean	min	max	mean
abs-001	51198	11.0	3.5	156347	5.0	2.2
abs-002	24060	42.0	11.0	47079	46.0	10.0
abs-007	244765	28.0	4.2	1103267	38.0	4.9
cpf-001	176174	9.7	4.4	272194	7.8	3.6
das-010	20471	2.7	1.7	247519	1.7	1.3
edf-003	188957	7.1	2.9	3039117	6.6	3.3
edf-004	117423	9.6	4.3	1160885	7.9	4.1
edf-008	23081	5.4	2.5	107578	5.8	2.8
edf-010	158968	6.8	3.8	467496	4.6	2.9
edf-011	51247	8.1	3.9	115342	8.6	4.4
elf-001	4518	76.0	6.9	132526	81.0	6.4
jbd-001	19887	59.0	12.0	78229	68.0	10.0
las-001	27061	14.0	4.9	177843	13.0	4.1
las-003	48993	15.0	4.9	298669	9.2	3.6
las-005	51921	16.0	3.8	732352	6.5	2.9
mhi-302	17361	28.2	5.5	146210	8.2	2.2
mhi-303	23736	8.0	3.0	161767	5.67	2.2
par-001	294944	67.0	12.0	1152663	29.0	5.9
par-006	22735	128.0	17.0	736905	33.0	7.7
sof-001	17018	7.8	3.0	248011	5.7	1.8

Table 3. Heuristic DDS: BDD sizes and total numbers of nodes involved in the computation

	|BDD|			#nodes	
model	min	max	mean	min	max	mean
abs-001	1.2	8.1	3.1	1.0	3.4	1.9
abs-002	0.8	4.5	2.3	0.8	5.2	2.4
abs-007	0.9	22.0	3.2	1.2	18.0	3.4
cpf-001	0.9	9.6	4.3	1.0	7.6	3.6
das-010	1.0	2.6	1.8	1.1	1.8	1.5
edf-003	0.7	5.5	2.2	0.7	2.7	1.7
edf-004	1.0	7.2	3.8	0.9	5.7	2.9
edf-008	1.3	5.2	2.6	1.4	5.4	2.6
edf-010	1.3	7.7	4.2	1.0	5.0	2.8
edf-011	2.0	7.9	4.7	2.1	8.6	5.0
elf-001	2.4	5.0	3.8	1.7	2.9	2.4
jbd-001	0.9	52	11.0	0.9	59.0	9.2
las-001	1.3	8.8	3.8	1.2	8.1	3.3
las-003	0.8	11.0	4.3	1.0	8.8	3.2
las-005	0.8	12.0	2.5	0.8	5.6	2.3
mhi-302	1.1	7.2	3.2	0.9	2.6	1.5
mhi-303	1.0	5.1	2.6	0.9	3.2	1.8
par-001	1.2	61.0	11.0	1.0	21.0	5.1
par-006	1.4	77.0	15.0	1.8	18.0	6.0
sof-001	1.0	7.7	2.9	1.0	5.6	1.7

Table 4. Heuristic DUS: BDD sizes and total numbers of nodes involved in the computation

	|BDD|			#nodes	
name	min	max	mean	min	max	mean
abs-001	3.6	3.6	3.6	3.1	3.1	3.1
abs-002	0.6	1.0	0.8	1.0	1.5	1.3
abs-007	2.4	3.2	2.8	4.9	6.4	5.7
cpf-001	1.4	2.8	2	1.3	2.7	1.8
das-010	1.1	1.5	1.3	0.8	0.9	0.9
edf-003	2.2	2.3	2.3	1.2	1.2	1.2
edf-004	1.7	1.8	1.7	2.5	2.6	2.6
edf-008	2.1	2.2	2.1	2.7	2.9	2.8
edf-010	1.4	1.6	1.5	1.6	1.9	1.8
edf-011	2.9	3.4	3.1	4.0	4.5	4.2
elf-001	0.6	0.7	0.7	0.4	0.5	0.4
jbd-001	2.1	2.2	2.1	1.0	1.0	1.0
las-001	5.4	5.5	5.4	5.2	5.2	5.2
las-003	2.8	2.9	2.8	3.5	3.5	3.5
las-005	2.0	2.1	2.1	1.8	1.8	1.8
mhi-302	4.8	5.2	5.0	1.9	2.0	2.0
mhi-303	1.9	2.1	2.0	1.6	1.7	1.6
par-001	1.8	2.0	1.9	1.6	1.7	1.6
par-006	0.4	0.4	0.4	0.6	0.6	0.6
sof-001	1.1	3.1	2.6	0.8	1.3	1.2

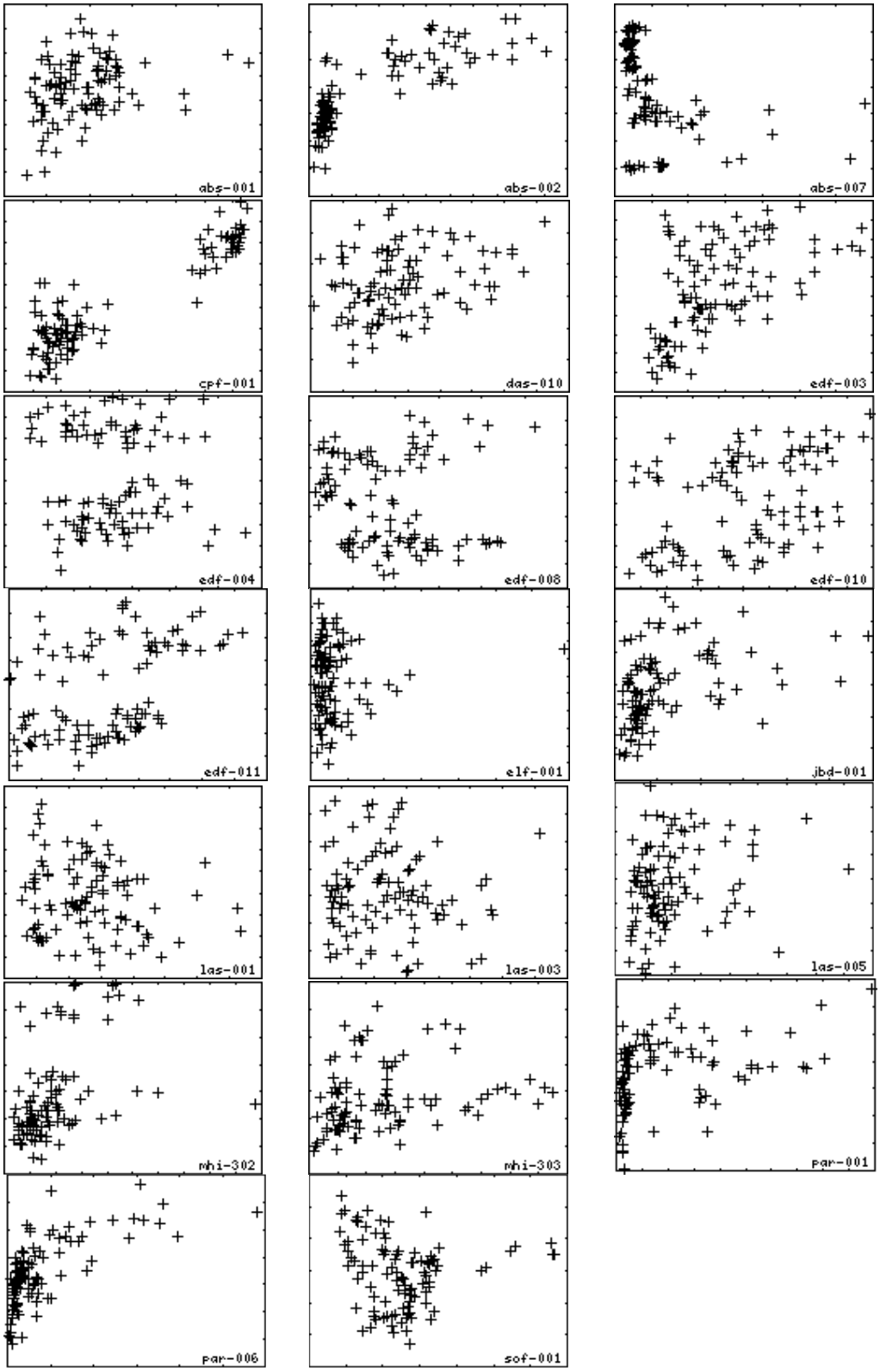

Figure 5. |BDD| (x-axis) versus ASAP graph widths (y-axis)

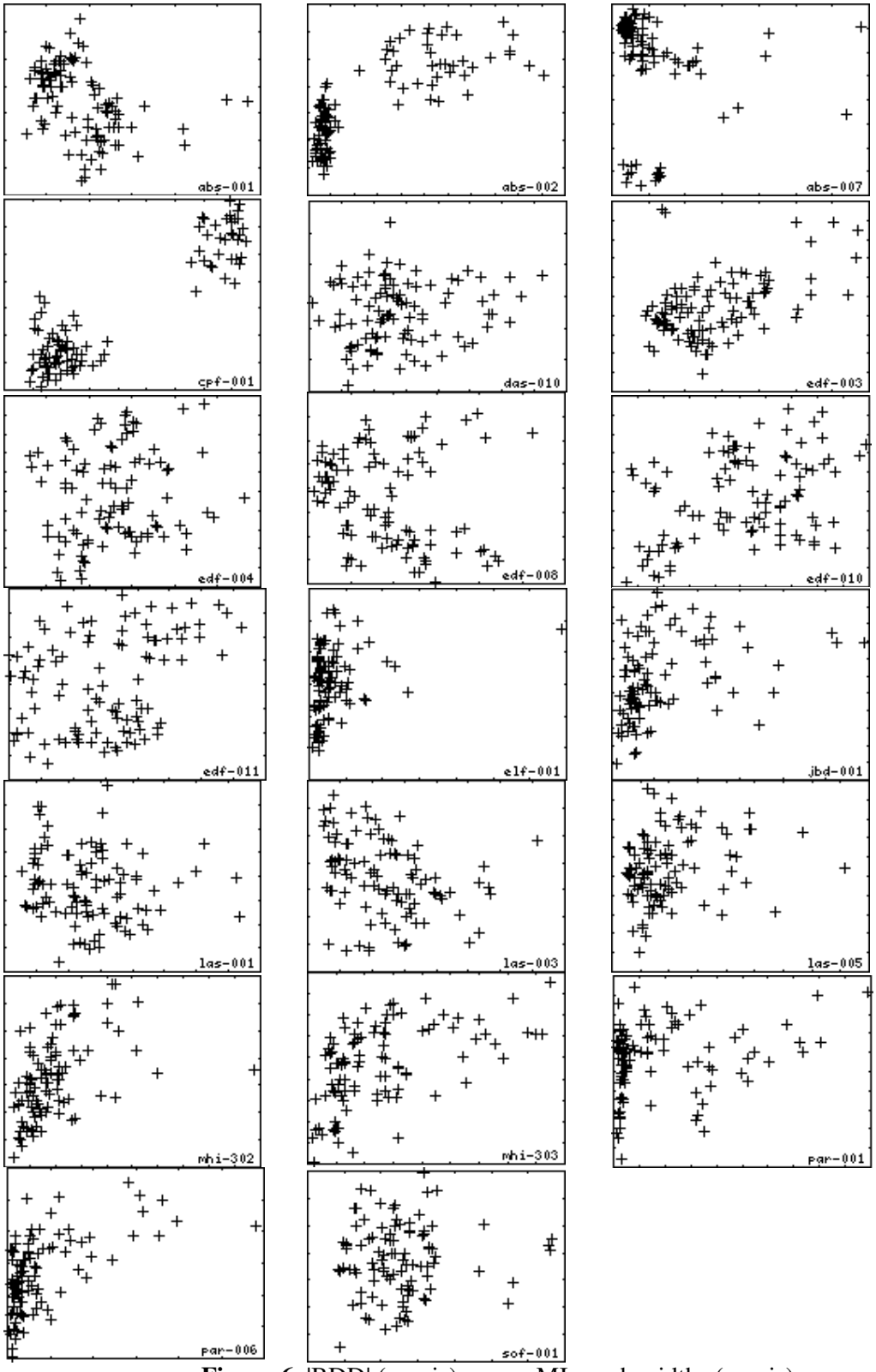

Figure 6. |BDD| (x-axis) versus MI graph widths (y-axis)